January 03, 2019

AUTOMATED FAILURE PREDICTION AND SELF-HEALING OF EDGE DEVICES IN INTERNET OF THINGS INFRASTRUCTURE

Lalit Jain

Nirup Pothireddy

Sai Kiran Reddy Malikireddy

Vivek Parekh

Bipin Algubelli

See next page for additional authors

Inventor(s)
Lalit Jain, Nirup Pothireddy, Sai Kiran Reddy Malikireddy, Vivek Parekh, Bipin Algubelli, Chandra Veluru, and Dileep Kumar

AUTOMATED FAILURE PREDICTION AND SELF-HEALING OF EDGE DEVICES IN INTERNET OF THINGS INFRASTRUCTURE

AUTHORS:
Lalit Jain
Nirup Pothireddy
Sai Kiran Reddy Malikireddy
Vivek Parekh
Bipin Algubelli
Chandra Veluru
Dileep Kumar

ABSTRACT

Techniques are described herein for predicting whether an Edge Device (ED) in the Internet of Things (IoT) infrastructure is going to crash or slowed using Prominent Feature Analysis (PFA) and supervised Machine Learning (ML) techniques like Support Vector Machines (SVM). This allows the IoT infrastructure administrator to remediate before a device crash or slow down.

DETAILED DESCRIPTION

Prominent Feature Analysis (PFA) technique enables obtaining the set of most prominent metrics of an Edge Device (ED) out of a pool of tens and hundreds of metrics using autoencoders and k-means clustering. The live and historical metric (e.g., Central Processing Unit (CPU), memory, network, Java ED (e.g., Java Virtual Machine (JVM)), etc.) data are collected for each of the EDs. With the subset of most prominent features from PFA, a Support Vector Machines (SVM) model may be built with a positive sample when the ED was down (e.g., taking samples before the crash happened, such as thirty minutes before the crash) and negative samples when the same ED was operating properly. Once the Machine Learning (ML) model is built, a live stream of metric data for each ED with the estimated prominent feature may be tested if the ED has resource congestion or is running normally. If the ED is running slowly, the alert may be triggered so that an Internet of Things (IoT) infrastructure administrator can remediate to avoid the failover or system slowness and/or restart the ED to avoid system crash.

Figures 1(a) and 1(b) below show the metric data for an ED (e.g., VM or any other ED). Figure 1(b) shows true failure for the ED. Finding the fixed hard threshold for a true

failure/crash is a difficult problem and even if one has a hard threshold, the same threshold might not generalize across several thousands of EDs running in the IoT infrastructure. The example of Figure 1 contains only three metrics, but in reality the ED can have several hundreds of metrics.

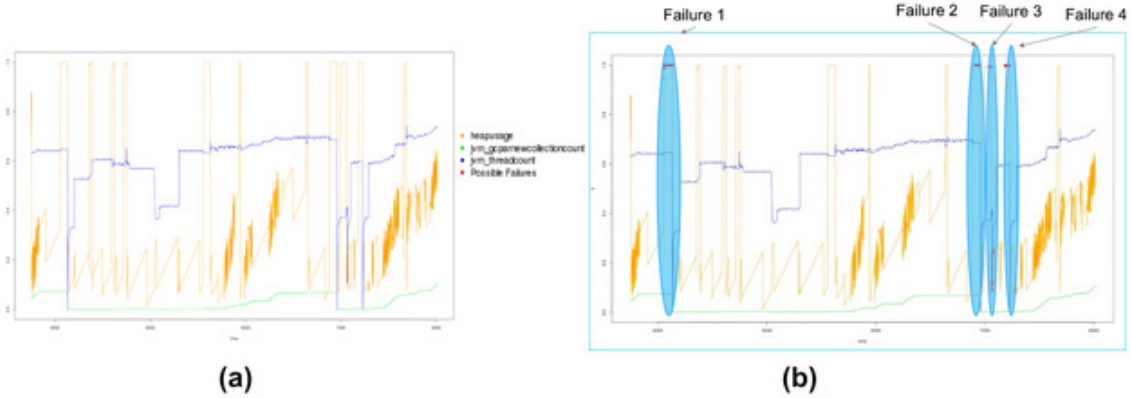

Figure 1: ED with three metrics (namely, heapusage, jvmgcparenwccolletioncount, and jvmthreadcount). 1(a) illustrates the guesses for a scenario when someone wants to choose the threshold of an individual metric for an ED crash. A specific threshold is a hard problem, as is whether the selected threshold can be generalized and applicable to other failures across numerous other EDs. 1(b) illustrates the ground truth of true failure on the same ED.

Thus a solution is necessary that scales/generalizes to thousands of EDs and prominent metrics which preserve essential information of the ED and not the noise in the data. The sub-selection of the metric that keeps most prominent information of the ED can be estimated using PFA. If the ground truth label and metric data for historical data when the ED was running normally and when it crashed is obtained, a supervised ML model may be built using these data. Once the ML model is built, it can be predicted whether an ED is going to crash given its live stream of metric data.

PFA is used to estimate the subset of most prominent features/metrics that are highly unique. SVM is popularly used to predict if the ED is going to crash. PFA may be performed by a single layer auto-encoder and using k-means clustering. The single layer auto-encoder projects the data into lower dimensions (i.e., a hidden node). Later the clustering is done for the weights across different features in the hidden node to find uncorrelated clusters and the feature that is the closest to mean of the cluster is chosen for the prominent feature(s). The PFA using the auto-encoder and k-means clustering works even if the features are not linearly separable in their input space. These techniques may be used for ED failure prediction and self-healing the IoT infrastructure.

As illustrated in Figure 2 below, the predictive ML solution consists of four components, namely the historical and live data collection unit, the feature selection and ML model building, the ED slowness or crash prediction, and the front end alert generation.

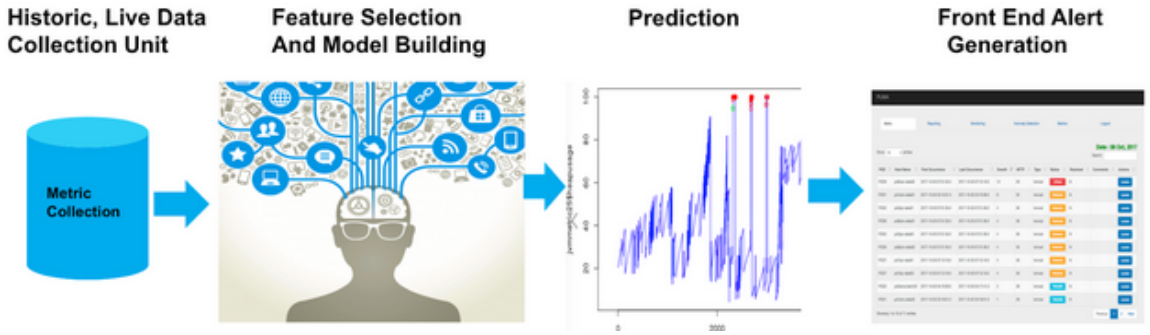

Figure 2: ML framework for ED failure prediction

With respect to the historical and live data collection unit, the EDs in the IoT infrastructure are registered to a monitoring system and time series database. The live metric data of all the EDs are collected from a monitoring system via real time streaming and stored in Hadoop Distributed File System (HDFS). Thus after a few days both historical and live metric data is present in the Hadoop system. Whenever the ED crashes due to resource contention, samples with all the metric data x minutes before the crash are collected as positive samples. The metric data for the same ED are collected when it was running normally as negative samples. Thus both the positive and negative samples for feature selection and ML model building are present.

With respect to feature selection and model building, the positive and negative metric samples are reviewed, and a subset of features that do not contribute to noise but possess the prominent information to represent the behavior of the ED are selected. The PFA is used to sub-select highly prominent metric data representing ED behavior. Let there be n metrics for each ED and m total positive and negative samples across all EDs in the specific category of the ED. Thus a metric of m x n dimensions may be obtained. Figure 3 below illustrates an example of single hidden layer auto-encoders.

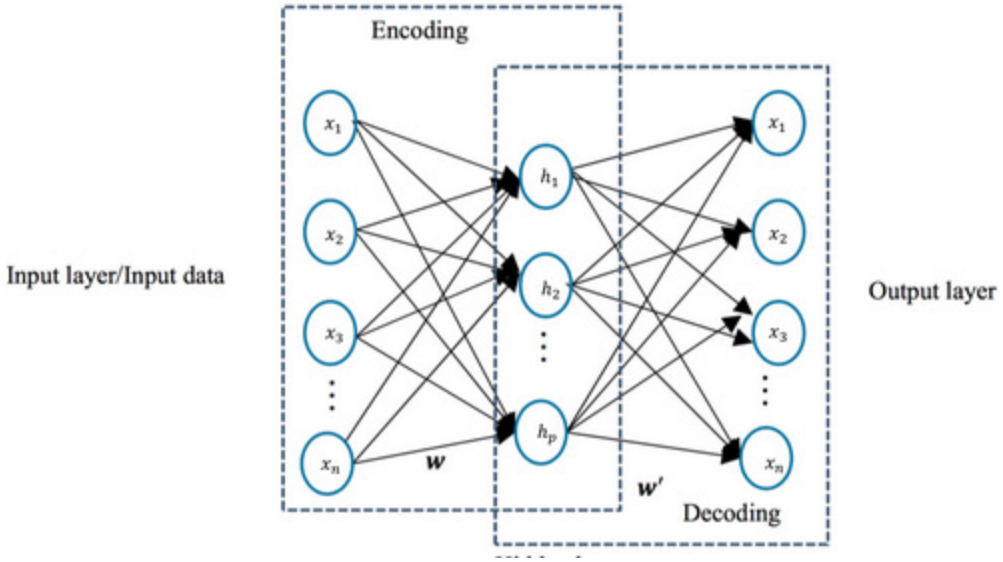

Figure 3: Auto-encoder with single hidden layer.

With respect to PFA, the following steps may be performed. First, the single layer auto-encoder may be run with m samples and n features. In a single layer auto-encoder, the network is trained to reconstruct its inputs at the output layer, which forces the hidden layer to try to learn a good representations of the inputs.

Second, the weight vectors across p hidden nodes are computed to form a p x m matrix.

$$\begin{array}{lllllll} h_1 = & w_{11} & w_{21} & w_{31} & \cdots & w_{n1} \\ h_2 = & w_{12} & w_{22} & w_{32} & \cdots & w_{n2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ h_p = & w_{1p} & w_{2p} & w_{3p} & \cdots & w_{np} \end{array}$$

Each principal component has a weight (w_*) assigned to each metric.

Third, the metrics which are highly correlated will have similar weights across columns. The idea here is to form the cluster of similar metrics. For this, the metric in the second step may be transposed and n samples obtained for the clustering algorithm as shown below.

$$\begin{array}{lllllll} s_1 = & w_{11} & w_{12} & w_{13} & \cdots & w_{1p} \\ s_2 = & w_{21} & w_{22} & w_{23} & \cdots & w_{2p} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ s_n = & w_{n1} & w_{n2} & w_{n3} & \cdots & w_{np} \end{array}$$

The k-means clustering algorithm may be run with n samples. For each cluster, the corresponding vector s_{i} may be found which is closest to the mean of the cluster as a prominent feature. This step may provide the choice of k uncorrelated options (i.e., prominent features).

Once the k prominent features are extracted from m features (where k <=m), the next step is to build the ML model with positive and negative labels. The positive and negative samples with k prominent features are to be trained with SVM to build the ML model. This model is later used to predict whether a specific ED is going to crash/slow down. Before training the SVM model, one needs to specify the kernel and optimal parameter for SVM. SVM may be used with a Radial Basis Function kernel that has two parameters, C and Gamma. The optimal C and Gamma are estimated by a grid search on the following values of C and Gamma using five-fold cross validation.

C="0.03125 0.125 0.5 2 8 32 128 512 2048 8192 32768"

Gamma="0.000031 0.000122 0.000488 0.001953 0.007812 0.03125 0.125 0.5 2 8 32"

The C and Gamma where classification accuracy is maximum on the training data are finally selected for building the SVM model and this model is saved for ED slowness or crash prediction.

The SVM model built in the previous step may be loaded in the memory. The live stream of metric data for all the EDs in the IoT infrastructure may be obtained every t seconds. The k prominent features of an individual ED at the same timestamp forms one sample. This sample may be tested against the SVM model if the ED is going to crash (test sample detected positive sample) or not (test sample detected negative sample).

With respect to front end alert generation, often there might be the case where a specific metric spikes up for a very small time due to the noise environment and the ED might be running normally. To avoid this scenario and any false positives, a counter may be used for each individual ED which tracks the health of the ED. For each ED, if the sample at test is predicted as a positive sample then the ED counter is incremented by one, and if the sample is detected as a negative sample then its counter is decremented by one. If at any point the ED counter is going negative, it is masked to zero and thus any ED counter is always greater than or equal to zero. The alert may be generated if the ED counter

is greater than C for any specific ED as shown in Figure 4 below. Event# is the ED/host counter.

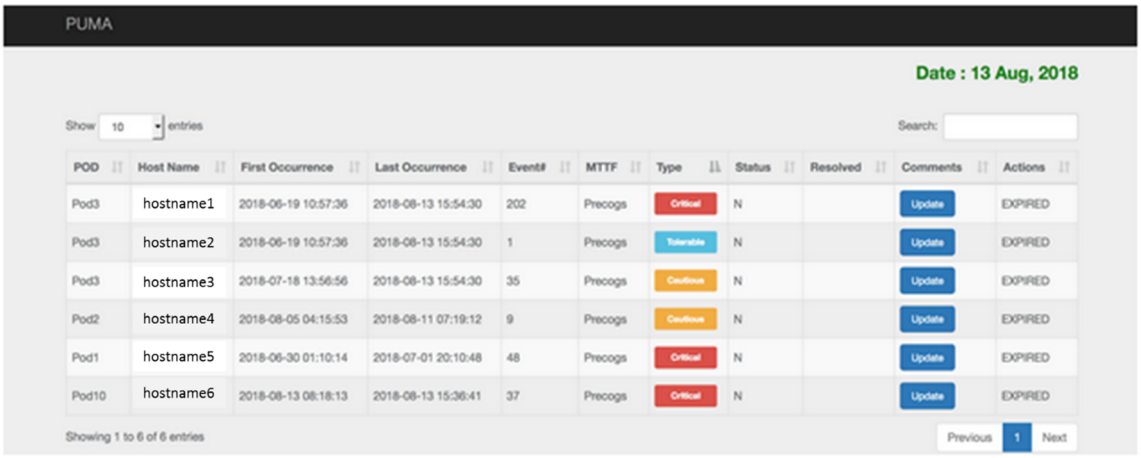

Figure 4: Front End Alert Generation User Interface

In summary, techniques are described herein for predicting whether an ED in the IoT infrastructure is going to crash or slowed using PFA and supervised ML techniques like SVM. This allows the IoT infrastructure administrator to remediate before a device crash or slow down.